Remarks

Reconsideration of this application as amended is respectfully requested.

Claims 1, 11, and 25 stand rejected under 35 U.S.C. §112, first paragraph.

Claims 1, 11, and 25 stand rejected under 35 U.S.C. §112, second paragraph.

Claims 1, 2 and 16 stand rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent Application Publication US 2001/0049686 of *Nelson et al.* ("*Nelson*").

Claims 3-6, 11, 17-20 and 25 stand rejected under 35 U.S.C. §103(a) as being unpatentable over *Nelson* and U.S. Patent no. 3,858,182 of *Delagi et al.* ("*Delagi*").

Claims 5 and 19 stand rejected under 35 U.S.C. §103(a) as being unpatentable over *Nelson* and U.S. Patent no. 5,619,710 of *Travis et al.* ("*Travis*").

Claims 6 and 20 stand rejected under 35 U.S.C. §103(a) as being unpatentable over *Nelson* and U.S. Patent no. 6,158,045 of *You* ("*You*").

Claims 7-10, 14-15, 21-24, and 28-29 stand rejected under 35 U.S.C. §103(a) as being unpatentable over *Nelson* and U.S. Patent no. 6,269,391 of *Gillespie* ("*Gillespie*").

Claims 12-13, and 26-27 stand rejected under 35 U.S.C. §103(a) as being unpatentable over *Nelson* and U.S. Patent no. 5,630,128 of *Farrell et al.* ("*Farrell*").

The Examiner has rejected claims 1, 11, and 25 under 35 U.S.C. §112, first paragraph, as containing subject matter not described in the Specification. The Examiner has stated that the "set of methods" recited in claim 1 and "set of structures" recited in claims 11 and 25 are not defined in the Specification.

Applicants respectfully submit that the term "set of methods" is defined in the Specification, e.g. page 10, lines 2-25 and page 11, lines 26-38. Nevertheless, applicants have amended claim 1 to recited "routine" in order to avoid

confusion with the word method as pertaining to a type of claim.

Applicants also submit that the term "set of structures" is defined in the Specification. For example, the Specification states that

> The following structure is used to define a user level thread in the context of the underlying platform for the software system 10. PTHREAD refers to a user level thread in terms of the underlying operating system 18.

```
typedef struct thread_t{
  Lock      lock;       /* Lock for this thread structure */
  pthread_t pthr_id;    /* The id of the corresp. PTHREAD */
  Utf8Const *name;      /* The thread name */
  JVM_word  state;      /* The tread state */
  Object    *thrObj;    /* Object corresp. to this thread */
  VmContext vm;         /* Ptr to Virt. Mach. Rntime Info */
  Thread_t  *nest;
} * Thread;
```

(Specification, page 12, lines 10-28) (Emphasis added) and that

> The method threadStop() is called to stop a specified thread and to clean up some of the structures associated with the specified thread.

```
int threadStop(Thread t)
{
Lock Thread Struct
Set state of t to DEAD
Resume all threads waiting on t.
Unlock
Lock the GTL
Remove t from the GTL
Unlock GTL
if (t == currentThread)
scheduler()
}
```

(Specification, page 15, lines 13-28) (Emphasis added) and that

> The NTIL 16 maintains a global thread list (GTL) which is a linked list of all user level threads. One global lock is used for adding or deleting thread structures from the GTL.

(Specification, page 12, lines 33-36) (Emphasis added).

The Examiner has rejected claims 1, 11 and 25 under 35 U.S.C. §112, second paragraph, as being incomplete for omitting essential elements which amounts to a gap between elements. The Examiner has stated that the omitted elements

are the "set of methods" in claim 1 and the "set of structures" in claims 11 and 25. As noted above, amended claim 1 recites a "routine" rather than a "set of methods." Applicants respectfully submit that no gap exists in amended claim 1 because the "native threads support routine" recited therein adapts the "threads interface layer" of a virtual machine to an underlying platform of a software system. In addition, the recitation of "set of structures" in amended claim 11 provides a further limitation on the native threads support routine of amended claim 1. Similarly, the recitation of "set of structures" in amended claim 25 provides a further limitation on the native threads support routine of amended claim 16.

The Examiner has rejected claims 1, 2 and 16 under 35 U.S.C. §103(a) as being obvious in view of *Nelson*. Applicants respectfully submit, however, that amended claim 1 is not obvious in view of *Nelson*. Amended claim 1 is a software system that includes the limitations

> virtual machine including a threads interface layer that provides a standard threads interface in the virtual machine for parallel execution of a plurality of software tasks which are adapted to the virtual machine;
>
> native threads interface layer that includes at least one native threads support routine used by the standard threads interface for adapting parallel execution of the software tasks on a platform which underlies the virtual machine.

(Amended claim 1) (Emphasis added).

Nelson does not disclose or suggest a native threads interface layer that includes a routine for adapting parallel execution of software tasks written for a standard threads interface of a virtual machine to an underlying platform as claimed in amended claim 1. Rather than adapt software tasks of a virtual machine to an underlying platform, *Nelson* discloses a only the threads support of an underlying platform - in this case the Solaris platform. (*Nelson*, page 3, right column - last two lines and page 5, left column - first two lines).

Given that claims 2-15 depend from amended claim 1, it is submitted that claims 2-15 are not obvious in view of *Nelson*.

Applicants also submit that amended claim 16 is not obvious in view of *Nelson*. Amended claim 16 includes limitations similar to the limitations discussed above in amended claim 1. Therefore, the remarks stated above with respect to amended claim 1 also apply to amended claim 16.

Given that claims 17-29 depend from amended claim 16, it is submitted that claims 17-29 are not obvious in view of *Nelson*.

The Examiner has rejected claims 3-6, 11, 17-20 and 25 under 35 U.S.C. §103(a) as being obvious in view of *Nelson* and *Delagi*. As shown above, amended claims 1 and 16 from which claims 3-6, 11, 17-20 and 25 depend are not obvious in view of *Nelson*. *Delagi* merely discloses processor registers (*Delagi*, col. 2, lines 9-21) rather than a native threads interface layer that includes a routine for adapting parallel execution of software tasks written for a standard threads interface of a virtual machine to an underlying platform as claimed in amended claims 3-6, 11, 17-20 and 25.

The Examiner has rejected claims 5 and 19 under 35 U.S.C. §103(a) as being obvious in view of *Nelson* and *Travis*. As shown above, amended claims 1 and 16 from which claims 5 and 19 depend are not obvious in view of *Nelson*. *Travis* discloses a system for invoking application programs on remote platforms (*Travis*, col. 2, lines 64-66 and col. 12, lines 19-22) rather than a native threads interface layer that includes a routine for adapting parallel execution of software tasks written for a standard threads interface of a virtual machine to an underlying platform as claimed in claims 5 and 19.

The Examiner has rejected claims 6 and 20 under 35 U.S.C. §103(a) as being obvious in view of *Nelson* and *You*. As shown above, amended claims 1 and 16 from which claims 6 and 20 depend are not obvious in view of *Nelson*. *You* discloses a software debugging system (*You*, col. 9, lines 15-26) rather

than a native threads interface layer that includes a routine for adapting parallel execution of software tasks written for a standard threads interface of a virtual machine to an underlying platform as claimed in claims 6 and 20.

The Examiner has rejected claims 7-10, 14-15, 21-24, and 28-29 under 35 U.S.C. §103(a) as being obvious in view of *Nelson* and *Gillespie*. As shown above, amended claims 1 and 16 from which claims 7-10, 14-15, 21-24, and 29 depend are not obvious in view of *Nelson*. *Gillespie* discloses virtual machine thread scheduling in a multi-processor system (*Gillespie*, col. 1, line 55 through col. 2 line 5) rather than a native threads interface layer that includes a routine for adapting parallel execution of software tasks written for a standard threads interface of a virtual machine to an underlying platform as claimed in claims 7-10, 14-15, 21-24, and 28-29.

The Examiner has rejected claims 12-13, and 26-27 under 35 U.S.C. §103(a) as being obvious in view of *Nelson* and *Farrell*. As shown above, amended claims 1 and 16 from which claims 12-13, and 26-27 depend are not obvious in view of *Nelson*. *Farrell* discloses a system for scheduling threads (*Farrell*, col. 2, lines 24-45) rather than a native threads interface layer that includes a routine for adapting parallel execution of software tasks written for a standard threads interface of a virtual machine to an underlying platform as claimed in claims 12-13, and 26-27.

It is respectfully submitted that in view of the amendments and arguments set forth above, the applicable objections and rejections have been overcome.

The Commissioner is authorized to charge any underpayment or credit any overpayment to Deposit Account No. 08-2025 for any matter in connection with this response, including any fee for extension of time, which may be required.

Respectfully submitted,

Date: 11-14-02 By: Paul H. Horstmann

Paul H. Horstmann
Reg. No.: 36,167

Version with Markings to Show Changes Made

1. A software system with a two tier arrangement for threads support, comprising:

virtual machine including a threads interface layer [having a set of methods] that provides [thread support in the virtual machine according to] a standard threads interface [associated with] in the virtual machine for parallel execution of a plurality of software tasks which are adapted to the virtual machine;

native threads interface layer that [provides a set of methods that] includes at least one native threads support routine used by the standard threads interface for adapting parallel execution of the software tasks on [adapt the methods of the threads interface layer to] a platform which underlies the virtual machine [software system].

3. The software system of claim 1, wherein the threads interface layer maintains a set of context information for each [of a set of threads in the software system] software task in terms of the virtual machine.

4. The software system of claim 3, wherein [the] each set of context information includes a value for each of a set of virtual machine registers associated with corresponding [thread] software task.

5. The software system of claim 1, wherein the native threads interface layer maintains a set of context information for each [of a set of threads in the software system] software task in terms of the platform.

6. The software system of claim 5, wherein [the] each set of context information includes a value for each of a set of processor registers associated with the [platform]

corresponding software task.

7. The software system of claim 1, wherein the native threads support routine [interface layer includes a method that] enables the threads interface layer to suspend a particular [thread] software task.

8. The software system of claim 1, wherein the native threads support routine [interface layer includes a method that] enables the threads interface layer to resume a particular [thread] software task.

9. The software system of claim 1, wherein the native threads support routine [interface layer includes a method that] enables the threads interface layer to wait for completion of a particular [thread] software task.

10. The software system of claim 1, wherein the native threads support routine [interface layer includes a method that] enables the threads interface layer to yield execution to another [thread] software task.

11. The software system of claim 1, wherein the native threads support routine [interface layer includes a method that] enables the threads interface layer to stop an execution of a particular thread and to clean up a set of structures associated with the particular [thread] software task.

12. The software system of claim 1, wherein the native threads support routine [interface layer includes a method that] enables the threads interface layer to set a priority of a particular [thread] software task.

13. The software system of claim 1, wherein the native threads support routine [interface layer includes a method

that] enables the threads interface layer to obtain a priority of a particular [thread] software task.

14. The software system of claim 1, wherein the native threads support routine [interface layer includes a method that] enables the threads interface layer to obtain an identifier of a currently executing [thread] software task.

15. The software system of claim 1, wherein the native threads support routine [interface layer includes a method that] enables the threads interface layer to select a [thread] software task for execution.

16. A method for providing threads support for a virtual machine in a software system, comprising the steps of:

providing a threads interface layer in the virtual machine [including a set of methods] that provides [thread support according to] a standard threads interface [associated with] in the virtual machine for parallel execution of a plurality of software tasks which are adapted to the virtual machine;

providing a native threads interface layer [having a set of methods that adapt the methods of the threads interface layer to] that includes at least one native threads support routine used by the standard threads interface for adapting parallel execution of the software tasks on a platform which underlies the software system.

17. The method of claim 16, wherein [the methods in] the threads interface layer [perform the step of maintaining] maintains a set of context information for each [of a set of threads in the software system] software task in terms of the virtual machine.

18. The method of claim 17, wherein [the step of

maintaining a] each set of context information comprises [the step of maintaining] a value for each of a set of virtual machine registers associated with a corresponding [thread] software task.

19. The method of claim 16, wherein [the methods in] the native threads interface layer [perform the step of maintaining] maintains a set of context information for each [of a set of threads in the software system] software task in terms of the platform.

20. The method of claim 19, wherein [the step of maintaining a] each set of context information comprises [the step of maintaining] a value for each of a set of processor registers associated with the platform.

21. The method of claim 16, wherein the [methods in the native threads interface layer include a method that] native threads support routine performs the step of suspending a particular [thread] software task.

22. The method of claim 16, wherein the [methods in the native threads interface layer include a method that] native threads support routine performs the step of resuming a particular [thread] software task.

23. The method of claim 16, wherein the [methods in the native threads interface layer include a method that] native threads support routine performs the step of waiting for completion of a particular [thread] software task.

24. The method of claim 16, wherein the [methods in the native threads interface layer include a method that] native threads support routine performs the step of yielding execution to another [thread] software task in response to a

request from [one or more of the methods in] the threads interface layer.

25. The method of claim 16, wherein the [methods in the native threads interface layer include a method that] native threads support routine performs the steps of stopping execution of a particular [thread] software task and cleaning up a set of structures associated with the particular [thread] software task.

26. The method of claim 16, wherein the [methods in the native threads interface layer include a method that] native threads support routine performs the step of setting a priority of a particular [thread] software task.

27. The method of claim 16, wherein the [methods in the native threads interface layer include a method that] native threads support routine performs the step of obtaining a priority of a particular [thread] software task.

28. The method of claim 16, wherein the [methods in the native threads interface layer include a method that] native threads support routine performs the step of obtaining an identifier of a currently executing [thread] software task.

29. The method of claim 16, wherein the [methods in the native threads interface layer include a method that] native threads support routine performs the step of selecting a [thread] software task for execution.